arXiv:1607.01759v2 [cs.CL] 7 Jul 2016

Bag of Tricks for Efficient Text Classification

Armand Joulin **Edouard Grave** **Piotr Bojanowski** **Tomas Mikolov**
Facebook AI Research
{ajoulin,egrave,bojanowski,tmikolov}@fb.com

Abstract

This paper proposes a simple and efficient approach for text classification and representation learning. Our experiments show that our fast text classifier `fastText` is often on par with deep learning classifiers in terms of accuracy, and many orders of magnitude faster for training and evaluation. We can train `fastText` on more th in less than ten minutes usi ticore CPU, and classify tences among 312K classe minute.

1 Introduction

Building good representations for text classification is an important task with many applications, such as web search, information retrieval, ranking and document classification (Deerwester et al., 1990; Pang and Lee, 2008). Recently, models based on neural networks have become increasingly popular for computing sentence representations (Bengio et al., 2003; Collobert and Weston, 2008). While these models achieve very good performance in practice (Kim, 2014; Zhang and LeCun, 2015; Zhang et al., 2015), they tend to be relatively slow both at train and test time, limiting their use on very large datasets.

At the same time, simple linear models have also shown impressive performance while being very computationally efficient (Mikolov et al., 2013; Levy et al., 2015). They usually learn word level representations that are later combined to form sentence representations. In this work, we propose an extension of these models to directly learn sentence representations. We show that by incorporating additional statistics such as using bag of n-grams, we reduce the gap in accuracy between linear and deep models, while being many orders of magnitude faster.

Our work is closely related to standard linear text classifiers (Joachims, 1998; ; l., 2008). 012), our moti es inspired by pervised word repre sen lov (2014), our a icated infer enc presentations easily reusable on different problems. We evaluate the quality of our model on two different tasks, namely tag prediction and sentiment analysis.

2 Model architecture

A simple and efficient baseline for sentence classification is to represent sentences as bag of words (BoW) and train a linear classifier, for example a logistic regression or support vector machine (Joachims, 1998; Fan et al., 2008). However, linear classifiers do not share parameters among features and classes, possibly limiting generalization. Common solutions to this problem are to factorize the linear classifier into low rank matrices (Schutze, 1992; Mikolov et al., 2013) or to use multilayer neural networks (Collobert and Weston, 2008; Zhang et al., 2015). In the case of neural networks, the information is shared via the hidden

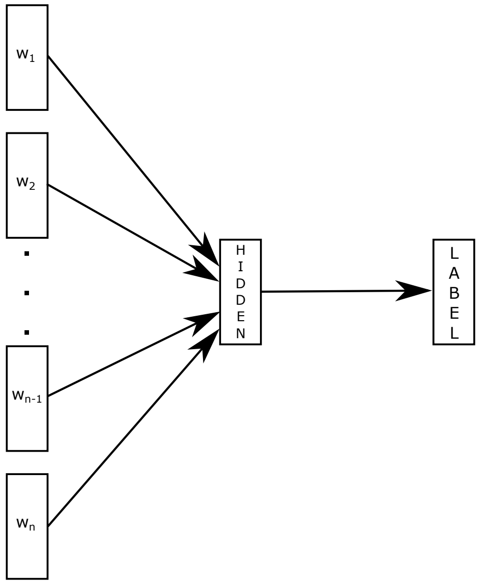

Figure 1: Model architecture for fast sentence classification.

layers.

Figure 1 shows a simple model with 1 hidden layer. The first weight matrix can b look-up table over the words of a s word representations are ave resentation, which is in turn fed t fier. This architecture is similar to the cbow model of Mikolov et al. (2013), where the middle word is replaced by a label. The model takes a sequence of words as an input and produces a probability distribution over the predefined classes. We use a softmax function to compute these probabilities.

Training such model is similar in nature to word2vec, i.e., we use stochastic gradient descent and backpropagation (Rumelhart et al., 1986) with a linearly decaying learning rate. Our model is trained asynchronously on multiple CPUs.

2.1 Hierarchical softmax

When the number of targets is large, computing the linear classifier is computationally expensive. More precisely, the computational complexity is $O(Kd)$ where K is the number of targets and d the dimension of the hidden layer. In order to improve our running time, we use a hierarchical softmax (Goodman, 2001) based on a Huffman coding tree (Mikolov et al., 2013). During training, the computational complexity drops to $O(d\log_2(K))$. In this tree, the targets are the leaves.

The hierarchical softmax is also advantageous at test time when searching for the most likely class. Each node is associated with a probability that is the probability of the path from the root to that node. If the node is at depth $l+1$ with parents $n_1, \ldots, n_l$, its probability is

$$P(n_{l+1}) = \prod_{i=1}^{l} P(n_i).$$

This means that the probability of a node is always lower than the one of its parent. Exploring the tree with a depth first search and tracking the maximum probability among the leaves allows us to discard any branch associated with a smaller probability. In practice, we observe a reduction of the complexity to $O(d\log_2(K))$ at test time. This approach is further extended to compute the T-top targets at the cost of $O(\log(T))$, using a binary heap.

2.2 N-gram features

rder but taking t is often compu- each we use bag of o capture some partial i der. This is ve ering comparab y use the order (Wang and Manning, 2012).

We maintain a fast and memory efficient mapping of the n-grams by using the *hashing trick* (Weinberger et al., 2009) with the same hashing function as in Mikolov et al. (2011) and 10M bins if we only used bigrams, and 100M otherwise.

3 Experiments

3.1 Sentiment analysis

Datasets and baselines. We employ the same 8 datasets and evaluation protocol of Zhang et al. (2015). We report the N-grams and TFIDF baselines from Zhang et al. (2015), as well as the character level convolutional model (char-CNN) of Zhang and LeCun (2015) and the very deep convolutional network (VDCNN) of Conneau et al. (2016). We also compare to Tang et al. (2015) following their evaluation protocol. We report their main baselines as well as

Model	AG	Sogou	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW (Zhang et al., 2015)	88.8	92.9	96.6	92.2	58.0	68.9	54.6	90.4
ngrams (Zhang et al., 2015)	92.0	97.1	98.6	95.6	56.3	68.5	54.3	92.0
ngrams TFIDF (Zhang et al., 2015)	92.4	97.2	98.7	95.4	54.8	68.5	52.4	91.5
char-CNN (Zhang and LeCun, 2015)	87.2	95.1	98.3	94.7	62.0	71.2	59.5	94.5
VDCNN (Conneau et al., 2016)	91.3	96.8	98.7	95.7	64.7	73.4	63.0	95.7
`fastText`, $h = 10$	91.5	93.9	98.1	93.8	60.4	72.0	55.8	91.2
`fastText`, $h = 10$, bigram	92.5	96.8	98.6	95.7	63.9	72.3	60.2	94.6

Table 1: Test accuracy [%] on sentiment datasets. `FastText` has been run with the same parameters for all the datasets. It has 10 hidden units and we evaluate it with and without bigrams. For VDCNN and char-CNN, we show the best reported numbers without data augmentation.

	Zhang and LeCun (2015)		Conneau et al. (2016)			`fastText`
	small char-CNN*	big char-CNN*	depth=9	depth=17	depth=29	$h = 10$, bigram
AG	1h	3h	8h	12h20	17h	3s
Sogou	-	-	8h30	13h40	18h40	36s
DBpedia	2h	5h	9h	14h50	20h	8s
Yelp P.	-	-	9h20	14h30	23h00	15s
Yelp F.	-	-	9h40	15h	1d	18s
Yah. A.	8h	1d	20h	1d7h	1d17h	27s
Amz. F.	2d	5d	2d7h	3d15h	5d20h	33s
Amz. P.	2d	5d	2d7h	3d16h	5d20h	52s

Table 2: Training time on senti rt the overall training time, except for char-CNN where we rep

their two approaches based on recurrent networks (Conv-GRNN and LSTM-GRNN).

Model	Yelp'13	Yelp'14	Yelp'15	IMDB
SVM+TF	59.8	61.8	62.4	40.5
CNN	59.7	61.0	61.5	37.5
Conv-GRNN	63.7	65.5	66.0	42.5
LSTM-GRNN	65.1	67.1	67.6	45.3
`fastText`	64.2	66.2	66.6	45.2

Table 3: Comparision with Tang et al. (2015). The hyper-parameters are chosen on the validation set.

Results. We present the results in Figure 1. We use 10 hidden units and run `fastText` for 5 epochs with a learning rate selected on a validation set from $\{0.05, 0.1, 0.25, 0.5\}$. On this task, adding bigram information improves the performance by $1 - 4\%$. Overall our accuracy is slightly better than char-CNN and a bit worse than VDCNN. Note that we can increase the accuracy slightly by using more n-grams, for example with trigrams, the performance on Sogou goes up to

for 97.1%. Finally, Figure 3 titive with the met 5). We tune the hyper-parameters on the validation set and observe that using n-grams up to 5 leads to the best performance. Unlike Tang et al. (2015), `fastText` does not use pre-trained word embeddings, which can be explained the 1% difference in accuracy.

Training time. Both char-CNN and VDCNN are trained on a NVIDIA Tesla K40 GPU, while our models are trained on a CPU using 20 threads. Table 2 shows that methods using convolutions are several orders of magnitude slower than `fastText`. Note that for char-CNN, we report the time per epoch while we report overall training time for the other methods. While it is possible to have a $10\times$ speed up for char-CNN by using more recent CUDA implementations of convolutions, `fastText` takes less than a minute to train on these datasets. Our speed-up compared to CNN based methods increases with the size of the dataset, going up to at

Input	Prediction	Tags
taiyoucon 2011 digitals: individuals digital photos from the anime convention taiyoucon 2011 in mesa, arizona. if you know the model and/or the character, please comment.	#cosplay	#24mm #anime #animeconvention #arizona #canon #con #convention #cos **#cosplay** #costume #mesa #play #taiyou #taiyoucon
2012 twin cities pride 2012 twin cities pride parade	#minneapolis	#2012twincitiesprideparade **#minneapolis** #mn #usa
beagle enjoys the snowfall	#snow	#2007 #beagle #hillsboro #january #maddison #maddy #oregon **#snow**
christmas	#christmas	#cameraphone #mobile
euclid avenue	#newyorkcity	#cleveland #euclidavenue

Table 4: Examples from the validation set of YFCC100M dataset obtained with `fastText` with 200 hidden units and bigrams. We show a few correct and incorrect tag predictions.

least a $15,000\times$ speed-up.

3.2 Tag prediction

Dataset and baseline. To test scalability of our approach, further evaluation is carried on the YFCC100M dataset (Ni et al., 2015) which consists of almost 100M images w

tags. We focus on predicting th

the title and caption (we do not

remove the words and tags occurring less than 100 times and split the data into a train, validation and test set. The train set contains 91,188,648 examples (1.5B tokens). The validation has 930,497 examples and the test set 543,424. The vocabulary size is 297,141 and there are 312,116 unique tags. We will release a script that recreates this dataset so that our numbers could be reproduced. We report precision at 1.

We consider a frequency-based baseline which predicts the most frequent tag. We also compare with Tagspace (Weston et al., 2014), which is a tag prediction model similar to ours, but based on the Wsabie model of Weston et al. (2011). While the Tagspace model is described using convolutions, we consider the linear version, which achieves comparable performance but is much faster.

Results and training time. Table 5 presents a comparison of `fastText` and the baselines. We run `fastText` for 5 epochs and compare it to Tagspace for two sizes of the hidden layer, i.e., 50 and 200. Both models achieve a similar performance with a small hidden layer, but adding bigrams gives us a significant boost in accuracy. At test time, Tagspace needs to compute the scores for all the classes which makes it relatively slow, while our fast inference gives a significant speed-up when the number of classes is large (more than 300K here). Overall, we are more than an order of magnitude

uality. The

ore significant

Model	prec@1	Running time	
		Train	Test
Freq. baseline	2.2	-	-
Tagspace, $h = 50$	30.1	3h8	6h
Tagspace, $h = 200$	35.6	5h32	15h
`fastText`, $h = 50$	30.8	6m40	48s
`fastText`, $h = 50$, bigram	35.6	7m47	50s
`fastText`, $h = 200$	40.7	10m34	1m29
`fastText`, $h = 200$, bigram	45.1	13m38	1m37

Table 5: Prec@1 on the test set for tag prediction on YFCC100M. We also report the training time and test time. Test time is reported for a single thread, while training uses 20 threads for both models.

Table 4 shows some qualitative examples. `FastText` learns to associate words in the caption with their hashtags, e.g., “christmas” with “#christmas”. It also captures simple relations between words, such as “snowfall” and “#snow”. Finally, using bigrams also allows it to capture relations such as “twin cities” and “#minneapolis”.

4 Discussion and conclusion

In this work, we have developed `fastText` which extends word2vec to tackle sentence and document classification. Unlike unsupervisedly trained word vectors from word2vec, our word features can be averaged together to form good sentence representations. In several tasks, we have obtained performance on par with recently proposed methods inspired by deep learning, while observing a massive speed-up. Although deep neural networks have in theory much higher representational power than shallow models, it is not clear if simple text classification problems such as sentiment analysis are the right ones to evaluate them. We will publish our code so that the research community can easily build on top of our work.

References

[Bengio et al.2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language

[Collobert and Weston2008] R Weston. 2008. A unified arch guage processing: Deep neu task learning. In *ICML*.

[Conneau et al.2016] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*.

[Deerwester et al.1990] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*.

[Fan et al.2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *JMLR*.

[Goodman2001] Joshua Goodman. 2001. Classes for fast maximum entropy training. In *ICASSP*.

[Joachims1998] Thorsten Joachims. 1998. *Text categorization with support vector machines: Learning with many relevant features*. Springer.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

[Le and Mikolov2014] Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

[Levy et al.2015] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL*.

[McCallum and Nigam1998] Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *AAAI workshop on learning for text categorization*.

[Mikolov et al.2011] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011. Strategies for training large scale neural network language models. In *Workshop on Automatic Speech Recognition and Understanding*. IEEE.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Ni et al.2015] Karl Ni, Roger Pearce, Kofi Boakye, Brian Van Essen, Damian Borth, Barry Chen, and Eric Wang. 2015. Large-scale deep learning on the YFCC100M dataset. In *arXiv preprint arXiv:1502.03409*.

[Pang and Lee2008] Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*.

[Rumelhart et al.1986] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning internal representations by error-propagation. In *Par- g: Explorations in the Mi-* MIT Press.

92 Dimensions of *uting*.

[Tan ing Liu. 2 ted recurrent neural n on. In *EMNLP*.

[Wa nd Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*.

[Weinberger et al.2009] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *ICML*.

[Weston et al.2011] Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*.

[Weston et al.2014] Jason Weston, Sumit Chopra, and Keith Adams. 2014. #tagspace: Semantic embeddings from hashtags. In *EMNLP*.

[Zhang and LeCun2015] Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

[Zhang et al.2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.